

Package 'kml3d'

March 28, 2012

Type Package

Title K-means for joint Longitudinal data

Version 2.0

Date 2012-04-01

Author Christophe Genolini

Maintainer Christophe Genolini <genolini@u-paris10.fr>

Description KmL3D is an implementation of k-means specifically design to deal with joint trajectories (longitudinal data on several variable-trajectories). It provide facilities to deal with missing value, compute several quality criterion (Calinski and Harabatz, Ray and Turie, Davies and Bouldin) and propose a graphical interphace for chosing the 'best' number of clusters.

License GPL (>= 2)

Lazyload yes

URL <http://www.r-project.org>

Collate global.r distance3d.r clusterLongData3d.r kml3d.r

Depends methods,clv,rgl,misc3d,longitudinalData,kml

Encoding latin1

Repository CRAN

Date/Publication 2012-03-28 11:09:17

R topics documented:

kml3d-package	2
affectIndiv3d	4
calculTrajMean3d	5
choice	6
clusterLongData3d	8
ClusterLongData3d-class	10
dist3d	12
generateArtificialLongData3d	13
kml3d	15
parKml3d	17
plot,ClusterLongData3d	19
plot3d,ClusterLongData3d	21
plot3dPdf	22
Index	24

kml3d-package ~ Overview: *KmL3D*, *K-means for joint Longitudinal data* ~

Description

kml3d is a new implementation of k-means for longitudinal data (or trajectories). Here is an overview of the package.

Details

```

Package:    kml3d
Type:       Package
Version:    2.0
Date:       2012-04-01
License:    GPL (>= 2)
Lazyload:   yes
Depends:    methods,graphics,rgl,misc3d,longitudinalData, kml
URL:        http://www.r-project.org
URL:        http://christophe.genolini.free.fr/kml

```

Overview

To cluster data, kml3d go through three steps, each of which is associated to some functions:

1. Data preparation
2. Building "optimal" clusterization.

3. Exporting results
4. Visualizing and exporting 3D object

1. Data preparation

km13d works on object of class `ClusterLongData3d`. Data preparation therefore simply consists in transforming data into an object `ClusterLongData3d`. This can be done via function `clusterLongData3d` (`cld3d` in short) that converts a `data.frame` or an `array` into a `ClusterLongData3d`.

Working on several variables measured on different scales can give too much weight to one of the dimensions. So the function `scale` normalizes data.

Instead of working on real data, one can also work on artificial data. Such data can be created with `generateArtificialLongData3d` (`gald3d` in short).

2. Building "optimal" clustering

Once an object of class `ClusterLongData3d` has been created, the algorithm `km13d` can be run.

Starting with a `ClusterLongData3d`, `km13d` builds several `Partition`. A `Partition` object is a partition of trajectories into subgroups. It also contains some information like the percentage of trajectories contained in each group or some quality criterion (like the Calinski & Harabasz).

k-means is a "hill-climbing" algorithm. The specificity of this kind of algorithm is that it always converges towards a maximum, but one cannot know whether it is a local or a global maximum. It offers no guarantee of optimality.

To maximize one's chances of getting a quality `Partition`, it is better to execute the hill climbing algorithm several times, then to choose the best solution. By default, `km13d` executes the hill climbing algorithm 20 times.

To date, it is not possible to know the optimum number of clusters even if the calculation of some quality criterion can give some clues. `km13d` computes various of them.

In the end, `km13d` tests by default 2, 3, 4, 5 and 6 clusters, 20 times each.

3. Exporting results

When `km13d` has constructed some `Partition`, the user can examine them one by one and choose to export some. This can be done via function `choice`. `choice` opens a graphic window showing various information including the trajectories clustered by a specific `Partition`.

When some `Partition` has been selected (the user can select more than 1), it is possible to save them. The clusters are therefore exported towards the file `name-cluster.csv`. Criteria are exported towards `name-criteres.csv`. The graphs are exported according to their extension.

4. Visualizing and exporting 3D object

`km13d` also proposes tools to visualize the trajectories in 3D. `plot3d` using the library `rgl` to plot two variables according to time (either the all set of joint-trajectories, or just the mean joint-trajectories). Then the user can make the graphical representation turn using the mouse. `plot3dPdf` builds a `Triangles` object. These kind of object can be included in a pdf file using `saveTrianglesAsASY` and the software `asymptote`. Once again, it is possible to make the image in the pdf file move using the mouse -so the reader gets real 3D-.

How to get help?

For those who are not familiar with S4 programming: In S4 programming, each function can be adapted for some specific arguments.

- To get help on a function (for example plot), use: `?(plot)`.
- To get help on a function adapted to its argument (for example plot on argument ClusterLongData), used: `?“plot,ClusterLongData”`.

Examples

```
### 1. Data Preparation
myCld <- generateArtificialLongData3d(c(15,15,15))

### 2. Building "optimal" clusterization (with only 3 redrawings)
kml3d(myCld,nbRedrawing=3)

### 3. Exporting results
try(choice(myCld))

### 4. Visualizing in 3D
plot3d(myCld)
```

affectIndiv3d ~ *Function: affectIndiv3d* ~

Description

Given some longitudinal data (trajectories) and k clusters centers, `affectIndiv3d` affects each individual to the cluster whose center is the closest.

Usage

```
affectIndiv3d(traj, clustersCenter, distance = dist3d)
```

Arguments

<code>traj</code>	[array(numeric)]: longitudinal data. Each line is an individual, each column is a time measurement, each plan of the third dimension is for one variable.
<code>clustersCenter</code>	[array(numeric)]: cluster center. Each line is a cluster centers, each column is a time measurement, each plan of the third dimension is for one variable. .
<code>distance</code>	[numeric <- function(joint-trajectory, joint-trajectory)]: distance between an individual and a clusters centre.

Details

Given an array of clusters center `clustersCenter` (each plan of the first dimension is a cluster center, that is `clusterCenter[2,,]` is the second cluster center), the function `affectIndiv3d` affect each individual of the array `traj` to the closest clusters, according to distance.

`affectIndiv3d` used with `calculTrajMean3d` simulates one k-means 3D step.

Value

Object of class `Partition`.

Examples

```
#####
### affectIndiv

### Some trajectories
traj <- gald3d()["traj"]

### 4 clusters centers
center <- traj[runif(4,1,nrow(traj)),,]

### Affectation of each individual
part <- affectIndiv3d(traj,center)

#####
### K-means simulation (4 steps)
plot(longData3d(traj),partition(part))
for (i in 1:4){
  center <- calculTrajMean3d(traj,part)
  part <- affectIndiv3d(traj,center)
  plot(longData3d(traj),partition(part))
}
```

calculTrajMean3d ~ *Function: calculTrajMean3d* ~

Description

Given some joint longitudinal data and a cluster affectation, `calculTrajMean3d` computes the mean joint-trajectories of each cluster.

Usage

```
calculTrajMean3d(traj, clust,centerMethod=function(x){mean(x,na.rm=TRUE)})
```

Arguments

traj	[array(numeric)]: joint longitudinal data. Each line is an individual, each column is a time measurement, the third dimension is for variables.
clust	[vector(numeric)]: affectation of each individual.
centerMethod	[joint-trajectory <- function(array(numeric))]: function used to compute the clusters' centers.

Details

Given a vector of affectation to a cluster, the function `calculTrajMean3d` compute the "central" trajectory of each clusters. The "center" can be define using the argument `centerMethod`.

`affectIndiv3d` used with `calculTrajMean3d` simulates one k-means step.

Value

An array of dimension (k, t, v) with k number of groups, t number of time mesurement and v number of variables.

Examples

```
#####
### calculTrajMean3d

### Some LongitudinalData3d
traj <- gald3d()["traj"]

### A partition
part <- floor(runif(150,1,5))
plot(longData3d(traj),partition(part))

### Clusters center
(center <- calculTrajMean3d(traj,part))

#####
### K-means simulation (4 steps)
plot(longData3d(traj),partition(part))
for (i in 1:4){
  part <- affectIndiv3d(traj,center)
  center <- calculTrajMean3d(traj,part)
  plot(longData3d(traj),partition(part))
}
```

 choice

 ~ Function: choice ~

Description

choice lets the user choose some Partition he wants to export.

Usage

```
choice(object, typeGraph = "bmp")
```

Arguments

object	[ClusterLongData3d]: Object containing the trajectories and all the Partition found by kml3d .
typeGraph	[character] for every selected Partition , choice export some graphs. typeGraph set the format that will be used. Possible formats are the ones available for savePlot .

Details

choice is a function that lets the user see the [Partition](#) found by [kml3d](#). At first, choice opens a graphics window. On the left side, all the [Partition](#) contain in [Object](#) are plotted by a number (the number of cluster of the [Partition](#)). The level of the number is proportionnal to a quality criteria (like Calinski & Harabatz). One [Partition](#) is 'active', it is the one marked by a black dot.

On the right side, the trajectories of [Object](#) are drawn (one graph for each variable), according to the active [Partition](#).

From there, choice offers numerous options :

Arrow Change the active [Partition](#).

Space Select/unselect a [Partition](#) (the selected [Partition](#) are surrounded by a circle).

Return Export all the selected [Partition](#), then quit the function choice.

'e' Change the display (Trajectories alone / quality criterion alone / both)

'd' Change actif criterion.

'c' Sort the [Partition](#) according to the actif criterion.

'r' Change the trajectories' style.

'f' Change the means trajectories's style.

'g/t' Change the symbol size.

'y/h' Change the number of symbols.

When 'return' is pressed, the selected [Partition](#) are exported. Exporting is done in a specific named `objectName-Cx-y` where x is the number of cluster and y is the order in the sublist. Four files are created :

objectName-Cx-y-Clusters.csv Table with two columns. The first is the identifier of each trajectory (idAll); the second holds the cluster's affectation of the trajectory.

objectName-Cx-y-Detail.csv Table containing information about the clusteration (percentage of individual in each cluster, various qualities criterion, algorithm used to find the partition and convergence time.)

objectName-Cx-y-Traj.bmp Graph representing the trajectories. All the parameters set during the visualization (color of the trajectories, symbols used, mean color) are used for the export. Note that the 'typeGraph' argument can be used to export the graph in a format different than 'bmp'.

objectName-Cx-y-TrajMean.bmp Graph representing the means trajectories of each clusters. All the parameters set during the visualization (color of the trajectories, symbols used, mean color) are used for the export.

This four file are created for each selected Partition. In addition, two 'global' graphes are created :

objectName-criterionActif.bmp Graph presenting the values of the criterionActifall for all the Partition.

objectName-criterionAll.bmp For each cluster's number, the first Partition is considered. This graph presents on a single display the values of all the criterion for each first Partition. It is helpfull to compare the various qualities criterion.

Value

For each selected Partition, four files are saved, plus two global files.

See Also

Overview: [kml3d-package](#)
 Classes : [ClusterLongData3d](#), [Partition](#)
 Methods : [kml3d](#)
 Plot : [plot\(ClusterLongData\)](#)

Examples

```
### Creation of articficial data
cld1 <- gald3d()

### Clusterisation
kml3d(cld1,nbRedrawing=3,toPlot='both')

### Selection of the clustering we want
# (note that "try" is for compatibility with CRAN only,
# you probably can use "choice(cld1)")
try(choice(cld1))
```

clusterLongData3d ~ *Function: clusterLongData3d (or cld3d) ~*

Description

clusterLongData3d (or cld3d in short) is the constructor for [ClusterLongData3d](#) object.

Usage

```
clusterLongData3d(traj, idAll, time, timeInData, varNames, maxNA)
cld3d(traj, idAll, time, timeInData, varNames, maxNA)
```

Arguments

traj	[array(numeric)] or [data.frame]: structure containing the joint-trajectories. Each line (traj[i, ,]) is a joint-trajectory of an individual ; columns (traj[, j,]) refer to the time during which measures were made ; the third dimensions (traj[, ,l]) are for variables.
idAll	[vector(character)]: single identifier for each trajectory (ie each individual). Note that the identifiers are of type character (that allow to deal identifiers like XUK32-612, identifiers that our favorite epidemiologists are so good at providing). If idAll are numeric, they are converted into characters.
time	[vector(numeric)]: time at which measures were made.
timeInData	[list(vector(numeric))]: precise the column containing the trajectories. The list labels are the names of the variables (like list(A=c(2,3,4),B=c(5,7,9))).
varNames	[character]: name of the variable being measured.
maxNA	[numeric] or [vector(numeric)]: maximum number of NA that are tolerates on a trajectory. If a trajectory has more missing than maxNA, then it is remove from the analysis. Note the maxNA can take diffents values for each variable-trajectories. The default value is length(time)-2.

Details

clusterLongData3d construct a object of class `ClusterLongData`. Two cases can be distinguished:

traj is an array: the first dimension (line) are individual. The second dimension (column) are time at which the measurement are made. The third dimension are the differents variable-trajectories. For example, traj[, ,2] is the second variable-trajectory.

If idAll is missing, the individuals are labelled i1, i2, i3,...

If timeInData is missing, all the column are used (1:ncol(traj)).

If traj is a data.frame: lines are individual. Time of measurement and variables should be provide through timeInData. timeInData is a list. The label of the list are the variable-trajectories names. Elements of the list are the column containing the trajectories. For example, if timeInData=list(V=c(2,3,4),W=c(6,8,12)), then the first variable-trajectory is 'V', its mesearment are in column 2,3 and 4. The second variable-trajectory is 'W', its measurment are in column 6,8 and 12.

If idAll is missing, the first column of the data.frame is used.

Value

An object of class `ClusterLongData3d`.

Examples

```
#####
### Building an array
tr1n <- array(c(1,2,NA, 1,4,NA, 6,1,8, 10,NA,2, 3,NA,NA,
               4,NA,5, 6,3,4, 3,4,4, 4,NA,NA, 5,5,4),
              dim=c(3,5,2))
```

```
#####
### clusterLongData

### With maxNA=3
clusterLongData3d(traj=tr1n,
  idAll=as.character(c(100,102,104)),
  time=c(1,2,4,8,16),
  varNames=c("P","A"),
  maxNA=3
)

### With maxNA=2
### Individual 104 is exclude
clusterLongData3d(traj=tr1n,
  idAll=as.character(c(100,102,104)),
  time=c(1,2,4,8,16),
  varNames=c("P","A"),
  maxNA=2
)
```

ClusterLongData3d-class

~ Class: ClusterLongData3d ~

Description

ClusterLongData3d is an object containing joint-trajectories and associated [Partition](#).

Objects from the Class

[km13d](#) is an algorithm that builds a set of [Partition](#) from joint longitudinal data. ClusterLongData3d is the object containing the original joint longitudinal data and all the [Partition](#) that [km13d](#) finds.

When created, an ClusterLongData3d object simply contains initial data (the joint-trajectories). After the execution of [km13d](#), it contains the original data and the [Partition](#) which has just been find by [km13d](#).

Note that if [km13d](#) is executed several times, every new [Partition](#) are added to the original ones, no pre-existing [Partition](#) is erased.

Slots

`idAll` [vector(character)]: Single identifier for each of the joint-trajectory (each individual).
Usefull for exporting clusters.

`idFewNA` [vector(character)]: Restriction of `idAll` to the trajectories that does not have 'too many' missing value. See `maxNA` for details.

`time` [numeric]: Time at which measures are made.

varNames [vector(character)]: Names of the variable measured.

traj [array(numeric)]: Contains the joint longitudinal data. Each horizontal plan (first dimension) corresponds to the trajectories of an individual. Vertical plans (second dimension) refer to the time at which measures are made. Transversal plans (the third dimension) are for variables.

dimTraj [vector3(numeric)]: size of the array traj (ie c(length(idFewNA), length(time), length(varNames))).

maxNA [numeric] or [vector(numeric)]: Individual whose trajectories contain more missing value than maxNA are excluded from traj and will not be used in the analysis. Their identifier is preserved in idAll but not in idFewNA. When maxNA is a single number, it is used for all the variables.

reverse [matrix(numeric)]: contain the mean (first line) and the standard deviation (second line) used to normalize the data. Useful to restore the original data after a scaling operation.

criterionActif [character]: Store the criterion name that will be used by functions that need a single criterion (like [plotCriterion](#) or [ordered](#)).

initializationMethod [vector(character)]: list all the initialization method that has already been used to find some Partition (useful to not run several times a deterministic method).

sorted [logical]: are the Partition currently held in the object sorted in decreasing order ?

c1 [list(Partition)]: list of Partition with 1 clusters.

c2 [list(Partition)]: list of Partition with 2 clusters.

c3 [list(Partition)]: list of Partition with 3 clusters.

...

c26 [list(Partition)]: list of Partition with 26 clusters.

Extends

Class [LongData3d](#), directly. Class [ListPartition](#), directly.

Methods

object['xxx'] Get the value of the field xxx. Inherit from [LongData3d](#) and [ListPartition](#).

object['xxx']<-value Set the field xxx to value. xxx. Inherit from [ListPartition](#).

[plot](#) Display the ClusterLongData3d, one graph for each variable, according to a [Partition](#).

[plot3d](#) Display two variables of the ClusterLongData3d in 3D according to a [Partition](#).

[plot3dPdf](#) Export the AZY code for displaying two variables of the ClusterLongData3d in a 3D pdf graph.

Examples

```
### Building longData
traj <- array(c(1,2,3,1,4, 3,6,1,8,10, 1,2,1,3,2, 4,2,5,6,3, 4,3,4,4,4, 7,6,5,5,4),
             dim=c(3,5,2))

myCld <- clusterLongData3d(
  traj=traj,
  idAll=as.character(c(100,102,103)),
```

```

    time=c(1,2,4,8,15),
    varNames=c("P","A"),
    maxNA=3
)

### Show
myCld

### Get
myCld['varNames']

### Set
myCld['criterionActif']<-"Davies.Bouldin"

### Plot
plot(myCld)

```

dist3d

~ Function: dist3d ~

Description

Compute the distance between two joint trajectories.

Usage

```
dist3d(x, y, method = "euclidian", power = 2)
```

Arguments

x	[matrix(numeric)]: first trajectory. The column are time, the line are variables.
y	[matrix(numeric)]: second trajectory. The column are time, the line are variables.
method	[character]: method used. Should be one of the method used by the function dist .
power	[numeric]: if method="minkowski", power is the power used.

Details

Compute the distance between two joint trajectories, using one of the distance define by [dist](#).

Value

A numeric

Examples

```
### Generate artificial data
myCld <- gald3d()

### Distance between individual 1 and 3 (there are in the same group)
dist3d(myCld['traj'][1,,],myCld['traj'][3,,])

### Distance between individual 1 and 51 (there are in two different groups)
dist3d(myCld['traj'][1,,],myCld['traj'][51,,])
```

```
generateArtificialLongData3d
```

~ Function: generateArtificialLongData3d (or gald3d) ~

Description

This function build up an artificial longitudinal data set (joint trajectories) an turn them into an object of class [ClusterLongData](#).

Usage

```
gald3d(nbEachClusters=50,time=0:10,varNames=c("V","T"),
      functionClusters=list(function(t){c(0,0)},function(t){c(10,10)},function(t){c(10-t,10-t)}),
      constantPersonal=function(t){c(rnorm(1,0,2),rnorm(1,0,2))},
      functionNoise=function(t){c(rnorm(1,0,2),rnorm(1,0,2))},
      decimal=2,percentOfMissing=0)

generateArtificialLongData3d(nbEachClusters=50,time=0:10,varNames=c("V","T"),
                             functionClusters=list(function(t){c(0,0)},function(t){c(10,10)},function(t){c(10-t,10-t)}),
                             constantPersonal=function(t){c(rnorm(1,0,2),rnorm(1,0,2))},
                             functionNoise=function(t){c(rnorm(1,0,2),rnorm(1,0,2))},
                             decimal=2,percentOfMissing=0)
```

Arguments

nbEachClusters [vector(numeric)]: number of trajectories that each cluster must contain. If a single number is given, it is duplicated for all groups.

time [vector(numeric)]: time at which measures are made.

varNames [vector(character)]: names of the variables.

functionClusters [list(function)]: lists the functions that define the average trajectories of each cluster. Each functions shall return a vector containing one value for each variable of varNames.

constantPersonal [function] or [list(function)]: lists the functions defining the personal variation between an individual and the mean trajectories of its cluster. Note

that these function should be constant function (the personal variation can not evolve with time). If a single function is given, it is duplicated for all groups (see detail).

functionNoise	[function] or [list(function)]: lists the functions generating the noise of each trajectory within its own cluster. Each functions shall return a vector containing one value for each variable of varNames. If a single function is given, it is duplicated for all groups.
decimal	[numeric]: number of decimals used to round up values.
percentOfMissing	[numeric]: percentage (between 0 and 1) of missing data generated in each cluster. If a single value is given, it is duplicated for all groups. The missing values are Missing Completely At Random (MCAR).

Details

`generateArtificialLongData3d` (`gald3d` in short) is a function that construct a set of artificial joint longitudinal data. Each individual is considered as belonging to a group. This group follows a theoretical trajectory, function of time. These functions (one per group) are given via the argument `functionClusters`.

Within a group, the individual undergoes individual variations. Individual variations are given via the argument `functionNoise`.

The number of individuals in each group is given by `nbEachClusters`.

Finally, it is possible to add missing values randomly (MCAR) striking the data thanks to `percentOfMissing`.

Value

Object of class `ClusterLongData`.

Author

Christophe Genolini

1. UMR U1027, INSERM, Université Paul Sabatier / Toulouse III / France

2. CeRSME, EA 2931, UFR STAPS, Université de Paris Ouest-Nanterre-La Défense / Nanterre / France

References

[1] C. Genolini and B. Falissard
 "KmL: k-means for longitudinal data"
 Computational Statistics, vol 25(2), pp 317-328, 2010

[2] C. Genolini and B. Falissard
 "KmL: A package to cluster longitudinal data"
 Computer Methods and Programs in Biomedicine, 104, pp e112-121, 2011

See Also

[ClusterLongData3d](#), [clusterLongData3d](#), [generateArtificialLongData](#)

Examples

```
#####
### Default example

ex1 <- generateArtificialLongData3d()
plot3d(ex1,parTraj=parTRAJ(type="1"))
part1 <- partition(rep(1:3,each=50))
plot3d(ex1,part1,parTraj=parTRAJ(type="1"))

#####
### 4 lines with unbalanced groups

ex2 <- generateArtificialLongData3d(
  nbEachClusters=c(5,10,20,40),
  functionClusters=list(
    function(t)c(t,t^3/100),
    function(t)c(0,t),
    function(t)c(t,t),
    function(t)c(0,t^3/100)
  ),
  functionNoise = function(t){c(rnorm(1,0,1),rnorm(1,0,1))}
)
part2 <- partition(rep(1:4,time=c(5,10,20,40)))
plot3d(ex2,part2)
```

kml3d

~ Algorithm kml3d: K-means for Joint Longitudinal data ~

Description

kml3d is a new implementation of k-means for joint longitudinal data (or joint trajectories). This algorithm is able to deal with missing value and provides an easy way to re roll the algorithm several times, varying the starting conditions and/or the number of clusters looked for.

Here is the description of the algorithm. For an overview of the package, see [kml3d-package](#).

Usage

```
km13d(object, nbClusters = 2:6, nbRedrawing = 20, toPlot = "none",
  parAlgo = parKm13d())
```

Arguments

object	[ClusterLongData3d]: contains trajectories to clusterize and some Partition .
nbClusters	[vector(numeric)]: Vector containing the number of clusters with which kml3d must work. By default, nbClusters is 2:6 which indicates that kml3d must search partitions with respectively 2, then 3, ... up to 6 clusters. Maximum number of cluster is 26.
nbRedrawing	[numeric]: Sets the number of time that k-means must be re-run (with different starting conditions) for each number of clusters.
toPlot	[character]: during computation, kml3d can display some graphes. If toPlot="traj", then the trajectories are plot (like with function plot,ClusterLongData). If toPlot="criterion", the quality criterions are plot (like with function plotCriterion). If toPlot="both", the graphic windows is split in two and both graphs are displayed. If "none", there is no graphical display.
parAlgo	[ParKml]: set the option used by kml3d (like the starting condition, the imputation methods, the save frequency, the maximum number of iteration, , the distance used...) See ParKml for details. The default values are describe in parKml3d .

Details

kml3d works on object of class ClusterLongData. For each number i included in nbClusters, kml3d computes a [Partition](#) then stores it in the field cX of the object ClusterLongData according to its number of clusters 'X'. The algorithm starts over as many times as it is told in nbRedrawing. By default, it is executed for 2, 3, 4, 5 and 6 clusters 20 times each, namely 100 times.

When a Partition has been found, it is added to the slot $c1$, $c2$, $c3$, ... or $c26$. cX stores the all Partition with X clusters. Inside a sublist, the Partition are sorted from the biggest quality criterion to the smallest (the best are stored first, using [ordered,ListPartition](#)), or not.

Note that Partition are saved throughout the algorithm. If the user interrupts the execution of kml3d, the result is not lost. If the user run kml3d on an object, then running kml3d again on the same object will add some new Partition to the one already found.

The possible starting conditions are defined in [initializePartition](#).

Value

A [ClusterLongData3d](#) object, after having added some [Partition](#) to it.

Optimisation

Behind kml3d, there are two different procedures :

1. Fast: when the parameter distance is set to "euclidean3d" and toPlot is set to 'none' or 'criterion', kml3d call a C compiled (optimized) procedure.
2. Slow: when the user defines its own distance or if he wants to see the construction of the clusters by setting toPlot to 'traj' or 'both', kml3d uses a R non compiled programmes.

The C procedure is 25 times faster than the R one.

So we advice to use the R procedure 1/ for trying some new method (like using a new distance) or 2/ to "see" the very first clusters construction, in order to check that every thing goes right. Then it is better to switch to the C procedure (like we do in Example section).

If for a specific use, you need a different distance, feel free to contact the author.

See Also

Overview: [kml3d-package](#)
 Classes : [ClusterLongData3d](#), [Partition](#)
 Methods : [clusterLongData3d](#), [choice](#)

Examples

```
### Generation of some data
cld1 <- generateArtificialLongData3d(15)

### We suspect 2, 3, 4 or 5 clusters, we want 3 redrawing.
# We want to "see" what happen (so toPlot="both")
kml3d(cld1,2:5,3,toPlot="both")

### 3 seems to be the best.
# We don't want to see again, we want to get the result as fast as possible.
# Just, to check the overall process, we plot the criterion evolution
kml3d(cld1,3,10,toPlot="criterion")
```

parKml3d

~ Function: parKml3d ~

Description

parKml3d is a constructor of object [ParKml](#) that provide adequate default value for the use of function [kml3d](#).

Usage

```
parKml3d(saveFreq = 100, maxIt = 200, imputationMethod = "copyMean",
         distanceName = "euclidean3d", power = 2, distance = function() {
           }, centerMethod = meanNA, startingCond = "nearlyAll", nbCriterion =100)
```

Arguments

saveFreq	[numeric]: Long computations can take several days. So it is possible to save the object ClusterLongData3d on which works kml3d once in a while. saveFreq defines the frequency of the saving process. The ClusterLongData3d is saved every saveFreq clustering calculations. The object is saved in the file <code>objectName.Rdata</code> in the current folder.
maxIt	[numeric]: Set a limit to the number of iteration if convergence is not reached.

imputationMethod	[character]: the calculation of quality criterion can not be done if some value are missing. <code>imputationMethod</code> define the method use to impute the missing value. See imputation for detail.
distanceName	[character]: name of the distance used by k-means. If the <code>distanceName</code> is "euclidean3d", a compiled optimized version specifically design for joint-trajectories version is used. Otherwise, the function define in the slot <code>distance</code> is used.
power	[numeric]: If <code>distanceName="minkowski"</code> , this define the power that will be used.
distance	[numeric <- function(trajA, trajB)]: function that computes the distance between two trajectories. If no function is specified, the Euclidian distance with Gower adjustment (to deal with missing value) is used.
centerMethod	[numeric <- function(vector(numeric))]: k-means algorithm computes the centers of each cluster. It is possible to personalize the definition of "center" by defining a function "centerMethod". This function should take a vector of numeric as argument and return a single numeric -the center of the vector-.
startingCond	[character]: specifies the starting condition. Should be one of "randomAll", "randomK", "maxDist", "kmeans++", "kmeans+", "kmeans-" or "kmeans-" (see initializePartition for details). It also could take two specifics values: "all" stands for c("maxDist", "kmeans-") then an alternance of "kmeans-" and "randomK" while "nearlyAll" stands for "kmeans-" then an alternance of "kmeans-" and "randomK".
nbCriterion	[numeric]: set the maximum number of quality criterion that are display on the graph (since displaying a high criterion number an slow down the overall process, the default value is 100).

Details

`parKml3d` is a constructor of object [ParKml](#) that provide adequate default value for the use of function [kml3d](#).

Value

An object [ParKml](#).

Examples

```
### Generation of some data
cld1 <- generateArtificialLongData3d(c(15,15,15))

### Setting two different set of option :
(option1 <- parKml3d())
(option2 <- parKml3d(centerMethod=function(x)median(x,na.rm=TRUE)))

### Running kml. Formaly, the second exemple is 'k-median'
kml3d(cld1,4,1,toPlot="both",parAlgo=option1)
kml3d(cld1,4,1,toPlot="both",parAlgo=option2)
```

 plot,ClusterLongData3d

 ~ Function: plot for ClusterLongData3d ~

Description

plot the trajectories of an object [ClusterLongData3d](#) relatively to a [Partition](#). One graphe for each variable is displayed.

Usage

```
## S4 method for signature 'ClusterLongData3d,numeric'
plot(x,y,parTraj=parTRAJ(),parMean=parMEAN(),parWin=windowsCut(x['nbVar']),nbSample=200,toPlot=c("b
```

Arguments

x	[ClusterLongData3d]: Object containing the joint-trajectories to plot.
y	[numeric] or [vector2(numeric)]: Give the Partition to represent. If y is missing, the Partition with the highest quality criterion (the actif one) is selected. If y is a number, the first Partition of the sublist c-y is selected. If y is a couple of numeric, the y[2]th Partition of the sublist c-y[1] is selected.
parTraj	[ParLongData]: Specification of the plotting parameters of the individual trajectories. Fields that can be changes are 'type','col','pch','xlab' and 'ylab'. In addition to the standard possible values, the option col="clusters" can be use to color the individual trajectories according to their clusters (exemple: parTraj=parTRAJ(type="o",col="clusters")). See ParLongData for details.
parMean	[ParLongData]: Specification of the plotting parameters of the mean trajectories (only when y is non missing). Fields that can be changes are 'type','col','pch','pchPeriod' and 'cex'. See ParLongData for details.
parWin	[parWindin]: Set the graphical display of the windows. See ParWindows for details.
nbSample	[numeric]: Graphical display of huge sample can be time consuming. This parameters fixe the maximum number of trajectories (randomly chosen) that will be drawn.
toPlot	[character]: either 'traj' for plotting trajectories alone, 'criterion' for plotting criterion alone, 'both' for plotting both or 'none' for not display anything.
nbCriterion	[numeric]: if a single criterion is given to criterion (and thus is displayed for 'all' the Partition), this slot allows to fix a limit on the number of points that will be display.

Details

plot the trajectories of an object `ClusterLongData3d` relatively to the 'best' `Partition`, or to the `Partition` define by `y`.

Graphical option concerning the individual trajectory (`col`, `type`, `pch` and `xlab`) can be change using `parTraj`. Graphical option concerning the cluster mean trajectory (`col`, `type`, `pch`, `pchPeriod` and `cex`) can be change using `parMean`. For more detail on `parTraj` and `parMean`, see object of class `ParLongData`.

See Also

Overview: [kml3d-package](#)

Classes : `ClusterLongData3d`

Plot : `plot(longitudinalData)`, `plotCriterion`

Examples

```
#####
### Construction of the data

myCld <- gald3d()
part <- partition(rep(1:3,each=50))

### Basic plotting
plot(myCld)
plot(myCld,part)

#####
### Changing graphical parameters 'par'

### No letters on the mean trajectories
plot(myCld,part,parMean=parMEAN(type="l"))

### Only one letter on the mean trajectories
plot(myCld,part,parMean=parMEAN(pchPeriod=Inf))

### Color individual according to its clusters (col="clusters")
plot(myCld,part,parTraj=parTRAJ(col="clusters"))

### Mean without individual
plot(myCld,part,parTraj=parTRAJ(type="n"))

### No mean trajectories (type="n")
### Color individual according to its clusters (col="clusters")
plot(myCld,part,parTraj=parTRAJ(col="clusters"),parMean=parMEAN(type="n"))

### Only few trajectories
plot(myCld,part,nbSample=10,parTraj=parTRAJ(col='clusters'),parMean=parMEAN(type="n"))
```

 plot3d,ClusterLongData3d

 ~ Function: plot3d for ClusterLongData3d ~

Description

Plot two variables of a [ClusterLongData3d](#) object in 3D, optionnaly relatively to a [Partition](#).

Usage

```
## S4 method for signature 'ClusterLongData3d,missing'
plot3d(x,y,varY=1,varZ=2,parTraj=parTRAJ(type="n"),parMean=parMEAN(),nbSample=200)
## S4 method for signature 'ClusterLongData3d,numeric'
plot3d(x,y,varY=1,varZ=2,parTraj=parTRAJ(type="n"),parMean=parMEAN(),nbSample=200)
```

Arguments

x	[ClusterLongData3d]: Object containing the trajectories to plot.
y	[numeric] or [vector2(numeric)]: Define the Partition P that will be use to plot the object. P is a Partition hold in the field c2, c3, ... c26. If y=c(a,b), then P is the Partition number b with a clusters. If y=a, then P is the partition number 1 with a clusters. If y is missing, P is the Partition with the best criterion.
varY	[numeric] or [character]: either the number or the name of the first variable to display. 1 by default.
varZ	[numeric] or [character]: either the number or the name of the second variable to display. 2 by default.
parTraj	[ParLongData]: Set the graphical parameters used to plot the trajectories of the ClusterLongData3d. See ParLongData for details.
parMean	[ParLongData]: Set the graphical parameters used to plot the mean trajectories of each clusters ClusterLongData3d (only when y is non missing). See ParLongData for details.
nbSample	[numeric]: Graphical display of huge sample can be time consuming. This parameters fixe the maximum nombre of trajectories (randomly chosen) that will be drawn.

Details

Plot two variables of a [ClusterLongData3d](#) object in 3D. It use the [rgl](#) library. The user can make the graphical representation turn using its mouse.

See Also

[ClusterLongData3d](#)

Examples

```
#####
### Real example on array

time=c(1,2,3,4,8,12,16,20)
id2=1:120
f <- function(id,t)((id-1)%3-1) * t
g <- function(id,t)(id%2+1)*t
h <- function(id,t)(id%4-0.5)*(20-t)
myCld <- clusterLongData3d(array(cbind(outer(id2,time,f),outer(id2,time,g),outer(id2,time,h))+rnorm(120*8*3,0,3),
part <- partition(rep(1:6,20))

### Basic plot
plot(myCld,part)

### plot3d, variable 1 and 2
plot3d(myCld,part)

### plot3d, variable 1 and 3
plot3d(myCld,part,varZ=3)
plot3d(myCld,parTraj=parTRAJ(col="red"))
```

plot3dPdf

~ Function: plot3dPdf for ClusterLongData3d ~

Description

Given a [ClusterLongData3d](#) and a [Partition](#), this function create Triangle objects representing the 3D plot of two variables of the main trajectories.

Usage

```
## S4 method for signature 'ClusterLongData3d,missing'
plot3dPdf(x,y,varY=1,varZ=2)
## S4 method for signature 'ClusterLongData3d,numeric'
plot3dPdf(x,y,varY=1,varZ=2)
```

Arguments

x	[ClusterLongData]: Object containing the trajectories to plot.
y	[numeric]: Define Partition P that will be use to plot the object. P is a Partition hold in the field c2, c3, ... c26. If y=c(a,b), then P is the Partition number b with a clusters. If y=a, then P is the partition number 1 with a clusters. If y is missing, P is the Partition with the best criterion.
varY	[numeric] or [character]: either the number or the name of the first variable to display. 1 by default.
varZ	[numeric] or [character]: either the number or the name of the second variable to display. 2 by default.

Details

Create Triangle objects representing the 3D plot of the main trajectories of a [ClusterLongData](#).

The three functions [plot3dPdf](#), [saveTrianglesAsASY](#) and [makeLatexFile](#) are design to export a 3D graph to a Pdf file. The process is the following:

1. [plot3dPdf](#): Create a scene, that is a collection of Triangle object that represent a 3D images.
2. [saveTrianglesAsASY](#): Export the scene in an '.asy' file.
3. '.azy' can not be include in LaTeX file. LaTeX can read only '.pre' file. So the next step is to use asymptote to convert '.asy' tp '.pre'. This is done by the command `asy -inlineimage -tex pdfLatex scene.azy`.
4. The previous step did produce a file `scene+0.prc` that can be include in a LaTeX file. [makeLatexFile](#) create a LaTeX file that is directly compilable (using `pdfLatex`). It produce a pdf file that contain the 3D object.

Value

A Triangle object.

Author(s)

Christophe Genolini
INSERM U669 / PSIGIAM: Paris Sud Innovation Group in Adolescent Mental Health
Modal'X / Universite Paris Ouest-Nanterre- La Defense

Contact author : <genolini@u-paris10.fr>

References

Article "KmL: K-means for Longitudinal Data", in Computational Statistics, Volume 25, Issue 2 (2010), Page 317.

Web site: <http://christophe.genolini.free.fr/kml>

See Also

[makeTriangles](#)

Index

- *Topic **aplot**
 - plot3d, ClusterLongData3d, 21
- *Topic **chron**
 - choice, 6
 - kml3d, 15
 - kml3d-package, 2
 - plot, ClusterLongData3d, 19
- *Topic **classes**
 - ClusterLongData3d-class, 10
- *Topic **classif**
 - choice, 6
 - kml3d, 15
 - kml3d-package, 2
 - plot, ClusterLongData3d, 19
- *Topic **cluster**
 - choice, 6
 - generateArtificialLongData3d, 13
 - kml3d, 15
 - kml3d-package, 2
 - plot, ClusterLongData3d, 19
- *Topic **datagen**
 - generateArtificialLongData3d, 13
- *Topic **dplot**
 - kml3d, 15
 - kml3d-package, 2
 - plot, ClusterLongData3d, 19
- *Topic **iplot**
 - choice, 6
 - kml3d-package, 2
 - plot, ClusterLongData3d, 19
- *Topic **models**
 - kml3d-package, 2
- *Topic **nonparametric**
 - choice, 6
 - kml3d, 15
 - kml3d-package, 2
- *Topic **package**
 - kml3d-package, 2
 - plot3d, ClusterLongData3d, 21
- *Topic **robust**
 - kml3d, 15
 - kml3d-package, 2
- *Topic **spatial**
 - choice, 6
 - kml3d, 15
 - kml3d-package, 2
 - plot, ClusterLongData3d, 19
- *Topic **ts**
 - choice, 6
 - generateArtificialLongData3d, 13
 - kml3d, 15
 - kml3d-package, 2
 - plot, ClusterLongData3d, 19
 - plot3d, ClusterLongData3d, 21
 - [, ClusterLongData3d-method (ClusterLongData3d-class), 10
 - [, ParChoice-method (kml3d-package), 2
 - [<- , ClusterLongData3d-method (ClusterLongData3d-class), 10
 - [<- , ParChoice-method (kml3d-package), 2
 - affectIndiv3d, 4
 - calculTrajMean3d, 4, 5, 6
 - choice, 3, 6, 17
 - choice, ClusterLongData3d-method (choice), 6
 - choice-methods (choice), 6
 - cld3d, 3
 - cld3d (clusterLongData3d), 8
 - ClusterLongData, 9, 13, 14, 23
 - ClusterLongData3d, 3, 8, 9, 15–17, 19–22
 - clusterLongData3d, 3, 8, 15, 17
 - clusterLongData3d, ANY, ANY, ANY, ANY, ANY, ANY (clusterLongData3d), 8
 - clusterLongData3d, ANY, ANY, ANY, ANY, ANY, ANY-method (clusterLongData3d), 8
 - clusterLongData3d, missing, missing, missing, missing, missing, (clusterLongData3d), 8

